Colegiul Național “Ștefan cel Mare”

Tîrgu-Neamț

Joc de dame

Lucrare de atestat

Coordonator: Candidat:

Prof. Oancea Constantin Dumea Gabriel-Stelian

Clasa a XII-a RB

2021

Cuprins

[1 Python 3](#_Toc71551347)

[1.1 Prezentare generala a limbajului de cod 3](#_Toc71551348)

[1.2 Aplicații ale limbajului python 3](#_Toc71551349)

[1.2.a) Aplicatii web 3](#_Toc71551350)

[1.2.b) Inteligența Artificiala & Mechanical Learning 4](#_Toc71551351)

[1.2.c) Jocuri 4](#_Toc71551352)

[1.3 Pygame 5](#_Toc71551353)

[2 Dame 6](#_Toc71551354)

[2.1 Reguli de joc 6](#_Toc71551355)

[3 Algoritm 7](#_Toc71551356)

[3.1 Constante 7](#_Toc71551357)

[3.2 Piese 7](#_Toc71551358)

[3.3 Tabla de dame 9](#_Toc71551359)

[3.4 Main() 12](#_Toc71551360)

[4 Algoritmul mini-max 13](#_Toc71551361)

[4.1 Prezentarea teoretica a algoritmului 13](#_Toc71551362)

[4.2 Aplicarea algoritmului minimax în proiect 14](#_Toc71551363)

[5 Bibliografie 16](#_Toc71551364)

# Python

În acest paragraf voi prezenta limbajul de programare python si cateva domenii de aplicare ale acestuia

## Prezentare generala a limbajului de cod

*Python este un limbaj de programare dinamic multi-paradigmă, creat în 1989 de programatorul olandez Guido van Rossum. Van Rossum este și în ziua de astăzi un lider al comunității de dezvoltatori de software care lucrează la perfecționarea limbajul Python și implementarea de bază a acestuia.*

*Python este un limbaj multifuncțional folosit de exemplu de către companii ca Google sau Yahoo! pentru programarea aplicațiilor web, însă există și o serie de aplicații științifice sau de divertisment programate parțial sau în întregime în Python. Popularitatea în creștere, dar și puterea limbajului de programare Python au dus la adoptarea sa ca limbaj principal de dezvoltare de către programatori specializați și chiar și la predarea limbajului în unele medii universitare. Din aceleași motive, multe sisteme bazate pe Unix, inclusiv Linux, BSD și Mac OS X includ din start interpretatorul CPython.*

*Popularitatea limbajului se datorează faptului că permite crearea rapida a aplicatiilor care nu cer viteze inalte de procesare a datelor, dar si pentru ca este utilizat ca limbaj de criptare folosit în cadrul aplicațiilor scrise în alte limbaje.*

*În ceea ce privește paradigma de programare, Python poate servi ca limbaj pentru software de tipul object-oriented, dar permite și programarea imperativă, funcțională sau procedurală. Sistemul de tipizare este dinamic iar administrarea memoriei decurge automat prin intermediul unui serviciu „gunoier” (garbage collector). Alt avantaj al limbajului este existența unei ample biblioteci standard de metode(la momentul actual python are mai mult de 137000 de librarii)*

## Aplicații ale limbajului python

### Aplicatii web

*Când vine vorba de aplicații web, python este cel mai des utilizat pentru partea de backend.* *În dezvoltarea software, termenii frontend și backend se referă la separarea elementelor ce definesc interfața de prezentare a unei aplicații (front end a.k.a ce vede user-ul final) și nivelul de acces și redare date (backend a.k.a acțiunile intreprinse de server). Pentru partea de backend a paginilor web, python a dezvoltat o gama destul de larga de librarii cum ar fi “Flask”, ”Django”, “Pyramid”, “Web2py”, etc.*

### Inteligența Artificiala & Mechanical Learning

***Artificial Intelligence****(AI) este un domeniu ce se ocupă cu dezvoltarea de mașinării care sunt capabile să acționeze ca omul. În ce privește implementarea AI, Python deține un număr considerabil de module. Din lista de module se pot observa și subdomeniile AI:*

|  |  |
| --- | --- |
| * Mechanical Learning * Neural Network * Limbaj Natural & Procesare de Texte * Etc. | Python and Artificial Intelligence- Automate Intellect |

*Machine Learning (ML) este capacitatea unui computer de a ‘învăță’, fără a fi programat în mod specific, pe baza datelor care îi sunt oferite. În alte cuvinte, algoritmul este implementat o dată, iar computerul va putea face față unor noi seturi de date, pe care nu le-a mai întâlnit și pe care le va putea procesa*

### Jocuri

|  |
| --- |
| *Python nu este prea des menționat când vine vorba de crearea jocurilor datorita limbajelor precum Java, C++,C# care sunt mai dezvoltate in acest domeniu. Totusi python are posibilitatea de a crea joculete 2D de tip arcade cum ar fi “Snake”, “Mario”, “X&O”. Pantry crearea jocurilor in python se folosește libraria Pygame* |

## Pygame

*Pentru proiectul de atestat am decis sa fac un joc de dame in Pygame.*

*Pygame este un set de module proiectat pentru programarea jocurilor video. Ca nucleu, foloseste coduri optimizate din C(codurile in C sunt de 10-20 de ori mai rapide decat cele din python) si asamblari de coduri(care sunt de obicei de 100 de ori mai rapide). Pygame este extrem de portabil si ruleaza pe orice platforma si sistem de operare. Librăria este doar la o comanda distanță de instalare. Nu este nevoie să te chinui cu intalatul librăriei în afara packetelor găsite deja în sistemul de operare.*

*Libraria este simplă și ușor de utilizat. Copii și studenții învata în cursuri ușoare implementarea librariei, prin crearea de jocuri de tip shooter. Sunt în jur de 600 de jocuri create si publicate pe site-urile web dedicate librariei, joculete testate de mii de utilizatori. Nu este nevoie de un GUI(Graphic user interface—Interfață grafică). Poți folosi pygame doar dintr-o linie de comanda daca vrei doar să procesezi imagini, sa preiei inputul de la joystick sau să redai sunete. Nu necesită sute de mii de linii de cod pentru chestii pe care nu le vei folosi. Nucleul este ținut simplu, iar lucrurile extra precum librăriile GUI și efectele sunt realizate separat de pygame. Poți folosi diferite bucați din cod separate, sau poti folosi librarii de redare a sunetului pentru a îmbunătăți jocul. Multe dintre modulele librăriei pot fi inițializate și folosite separat.*

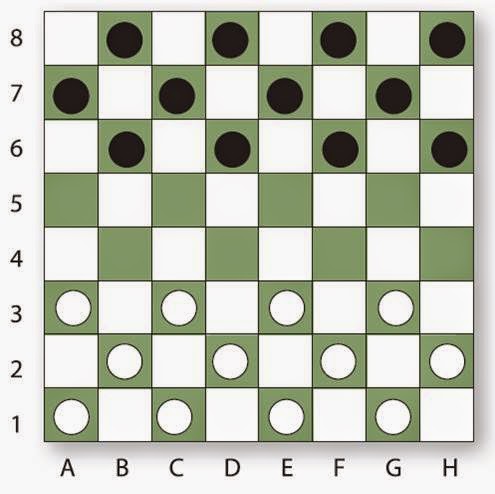


# Dame

*Dame este un joc pentru două persoane bazat pe strategia jucătorilor. Jocul este mai puțin răspândit în Germania, fiind foarte popular în schimb în Rusia și Olanda unde se organizează adevărate campionate.*

## Reguli de joc

*Jocul este de regulă alcătuit dintr-o tablă de șah cu 8 x 8 pătrate colorate alternativ deschis și întunecat. Figurile de joc sunt 24 de discuri rotunde confecționate din lemn, de diferite culori pentru fiecare jucator . Câmpul de joc îl constituie numai pătrățelele de culoare închisă, piesele de joc deplasându-se numai diagonal înainte, câte un pătrat, caz de excepție când poate sări peste piesa adversarului pe care o câștigă. Condiția de a putea sări piesa adversarului este ca în spatele piesei sărite să existe un câmp (pătrat) gol. Țelul jocului este de a ajunge pe ultimul rând de pătrățele din partea adversarului când piesele de joc suprapuse două câte două vor fi numite „dame”. Aceste piese duble de joc pot sări diagonal de-a lungul tablei. A câștigat jocul acea persoană care a câștigat toate piesele adversarului sau l-a blocat.*



# Algoritm

## Constante

*Pentru a îmi fi mai ușor să modific sau să mai adaug variabile, am salvat toate constantele într-o filă separata python. Am stabilit lățimea si lungimea tablei de joc ca fiind 800 de pixeli, si am creat 8 randuri și coloane. Pe urmă am stabilit dimensiunea fiecarui pătrățel de joc împărțind lățimea la numarul de coloane. Am creat variabile pentru fiecare culoare folosită în joc, folosind spectrul de culori RGB. De exemplu culoarea rosie are codul (255,0,0), alb are codul (255,255,255), negru (0,0,0), etc. Tot aici am prelucrat imaginea cu coroana, care urmează să fie folosită pentru a marca damele de pe tabla de sah.*

import pygame  
  
WIDTH, HEIGHT = 800, 800  
ROWS, COLS = 8, 8  
SQUARE\_SIZE = WIDTH//COLS  
  
RED = (255, 0, 0)  
WHITE = (255, 255, 255)  
BLACK = (0, 0, 0)  
BLUE = (0, 0, 255)  
GREY = (128,128,128)  
  
CROWN = pygame.transform.scale(pygame.image.load('assets/crown.png'), (44,25))

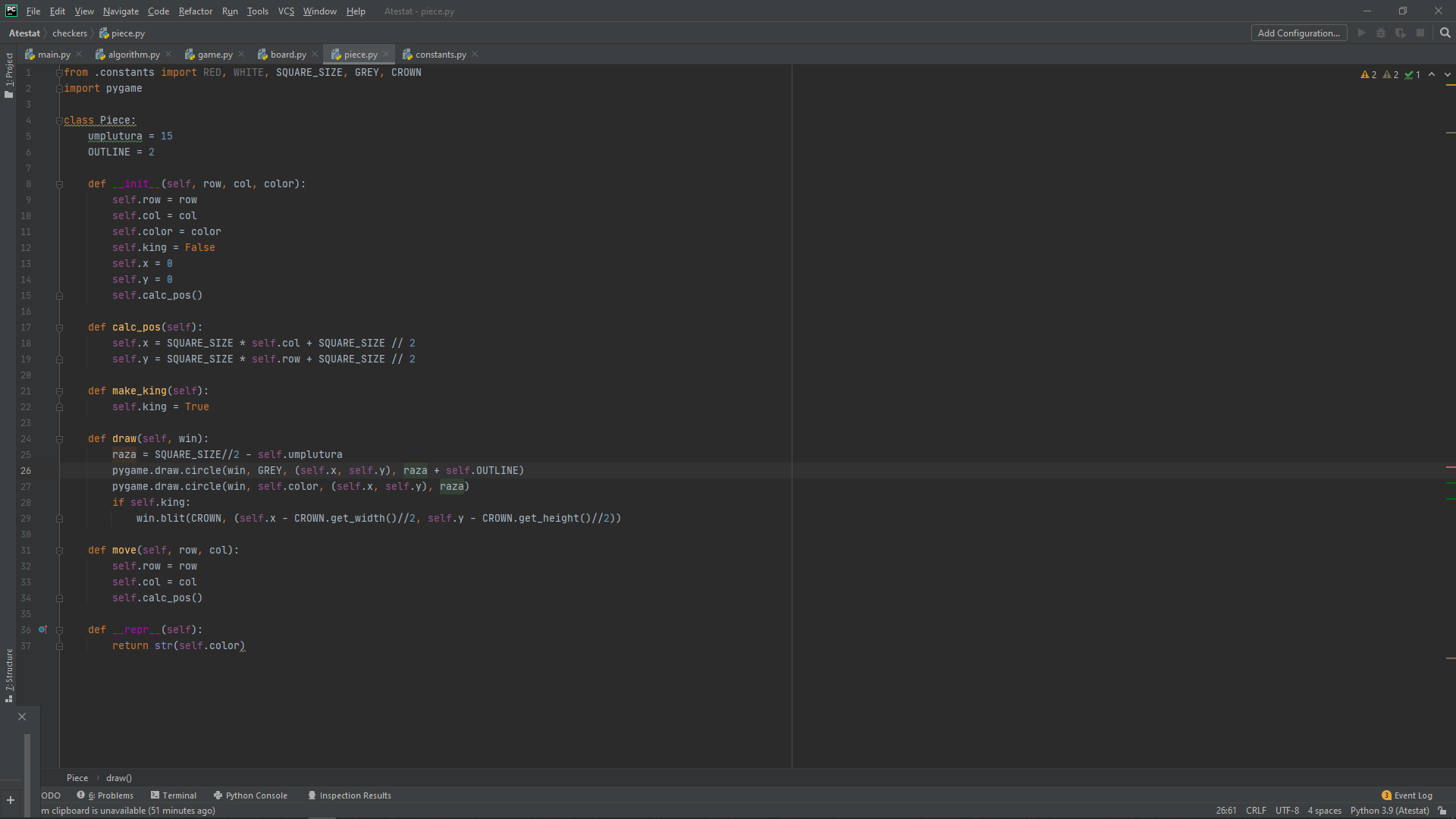
## Piese

*Piesele sunt create în fila piece.py. Din fila constante importăm culorile roșu, alb si gri, mărimea pătratului și coroana. Creăm clasa Piece ce fi folosită pentru a forma toate obiectele de tip piesă.*

*Variabila umplutură reține mărimea părții din pătrat care rămâne liberă, iar variabila outline reține marimea conturului fiecărei piese. În subprogramul \_\_init\_\_ care este specific tuturor claselor stabilim ca fiecare piesă să fie definită de rând, coloana, culoare, dacă este damă sau nu și de coordonatele x și y de pe tablă.*

*În subprogramul calc\_pos calculăm mijlocul fiecărui pătrățel pentru a putea desena fiecare piesa la mijlocul porțiunii unde se afla.*

*M-am ocupat de desenarea pieselor în subprogramul draw ce are ca parametru win ce reprezintă suprafața pe care se află piesele. Raza cercurilor vor fi egale cu jumătate din mărimea patratului din care scădem umplutura. Prima dată desenăm marginea cercului care se afla la distanță față de centru de o rază și marimea marginii. Apoi desenăm cercul propriuzis de rază “raza”. Stabilim daca piesa desenată este o dama, iar dacă este imprimam imaginea coroanei pe mijlocul cercului. Tot în această filă înlocuim pozițiile actuale ale pieselor cu cele pe care o să le stabilim mai în colo,*



## Tabla de dame

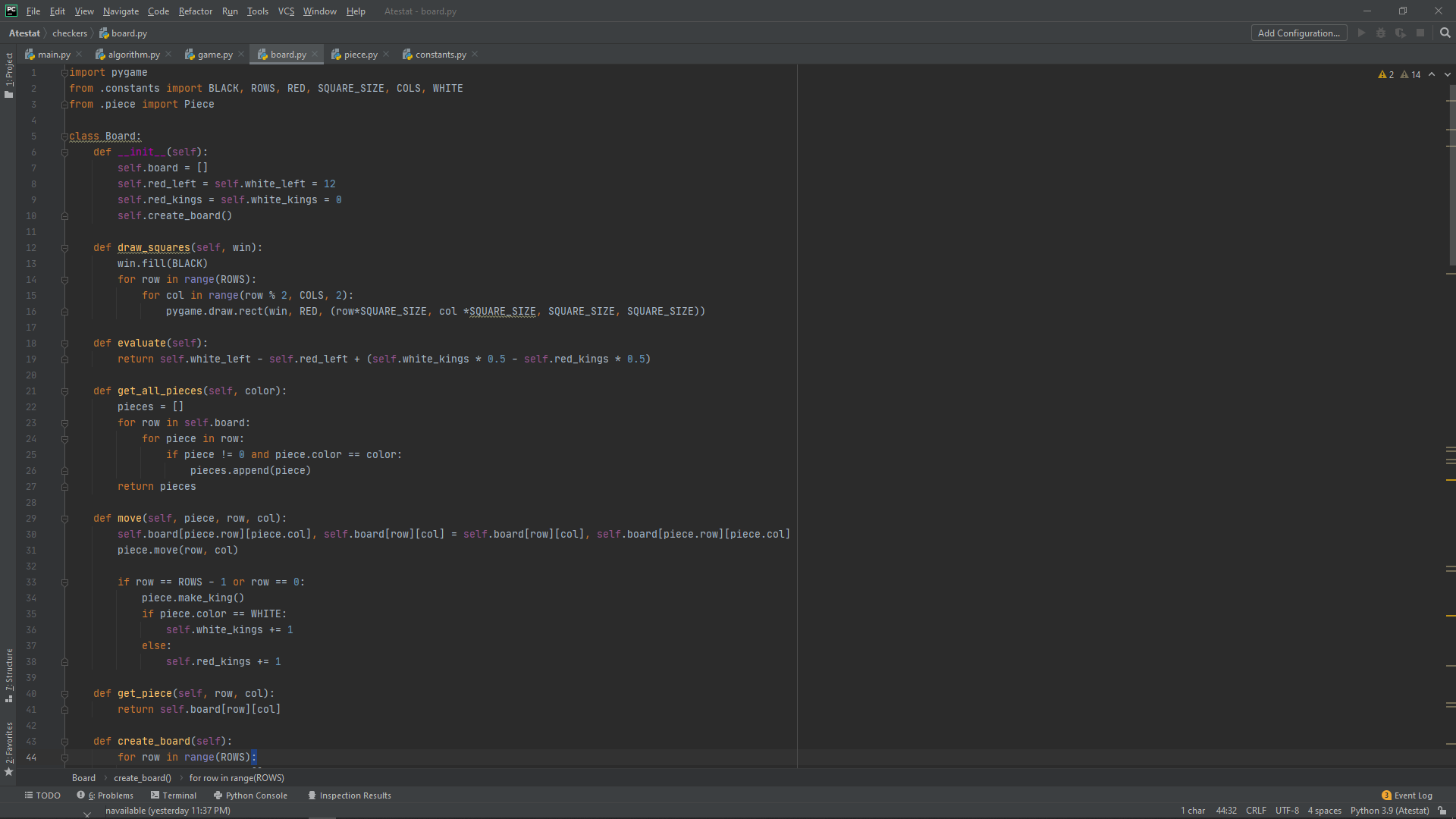
*Pentru a creea tabla de dame importăm culorile, coloanele si marimea pătratelor. De asemenea importăm piesele create anterior.*

*Pentru obiectele folosite de pe tabla de sah crăm clasa Board, obiectele self.board de tip lista ce va reține pozitiile fiecărei piese, self.red\_left si self.white\_left ce rețin câte piese de culorile respective mai sunt (inițial pe tablă fiind 12 piese de fiecare culoare) si self.create\_board ce va construi tabla de sah dupa fiecare activitate desfășurată.*

*Pentru a desena patratele folosesc subprogramul draw\_squares ce inițial creeaza o fereastră complet neagră. Pe urmă desenez pătratele de culoare roșie folosind functia draw.rect din pygame ce se folosește pentru a desena pătrate.*

*În subprogramul get\_all\_pieces preluam fiecare piesă si o integrăm în lista pieces pentru a ne fi mai ușor mai târziu să prelucrăm doar anumite piese.*

*Subprogramul move are rolul de a determina dacă o piesă a ajuns pe ultimul rând din partea opusă și transforma piesa in dama, dacă aceasta nu este deja damă.*

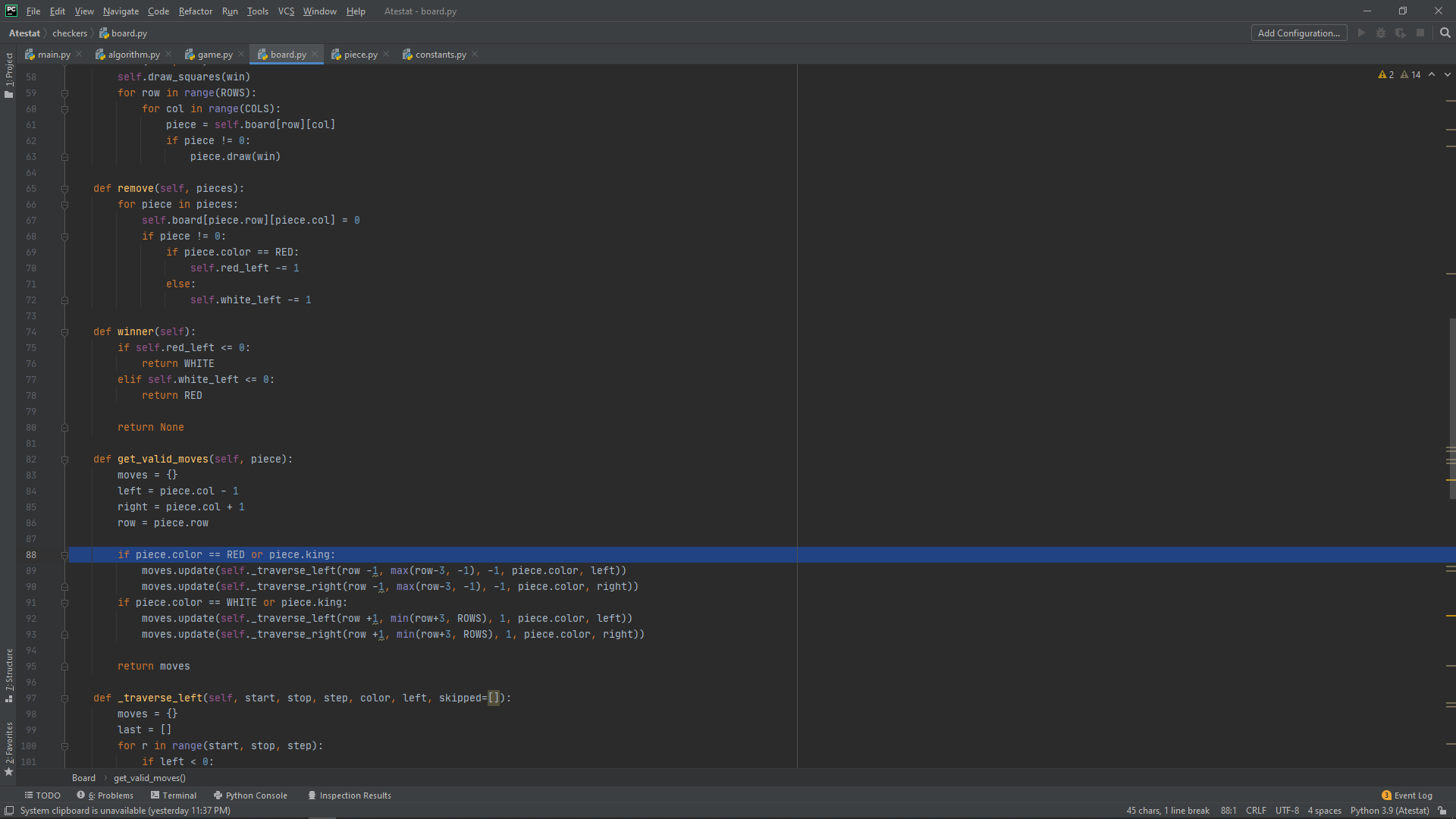


*Pentru a prelua pozitia unei piese am creeat subprogramul get\_piece ce returneaza randul si coloana piesei respective*

def get\_piece(self, row, col):  
 return self.board[row][col]

*Pentru a stabili câștigătorul subprogramul “winner” returnează culoarea alb daca nu mai sunt piese roșii ramase pe table, culoarea roșu pentru cazul contrar si nu returnează nimic dacă încă mai sunt piese ramase pe tablă.*

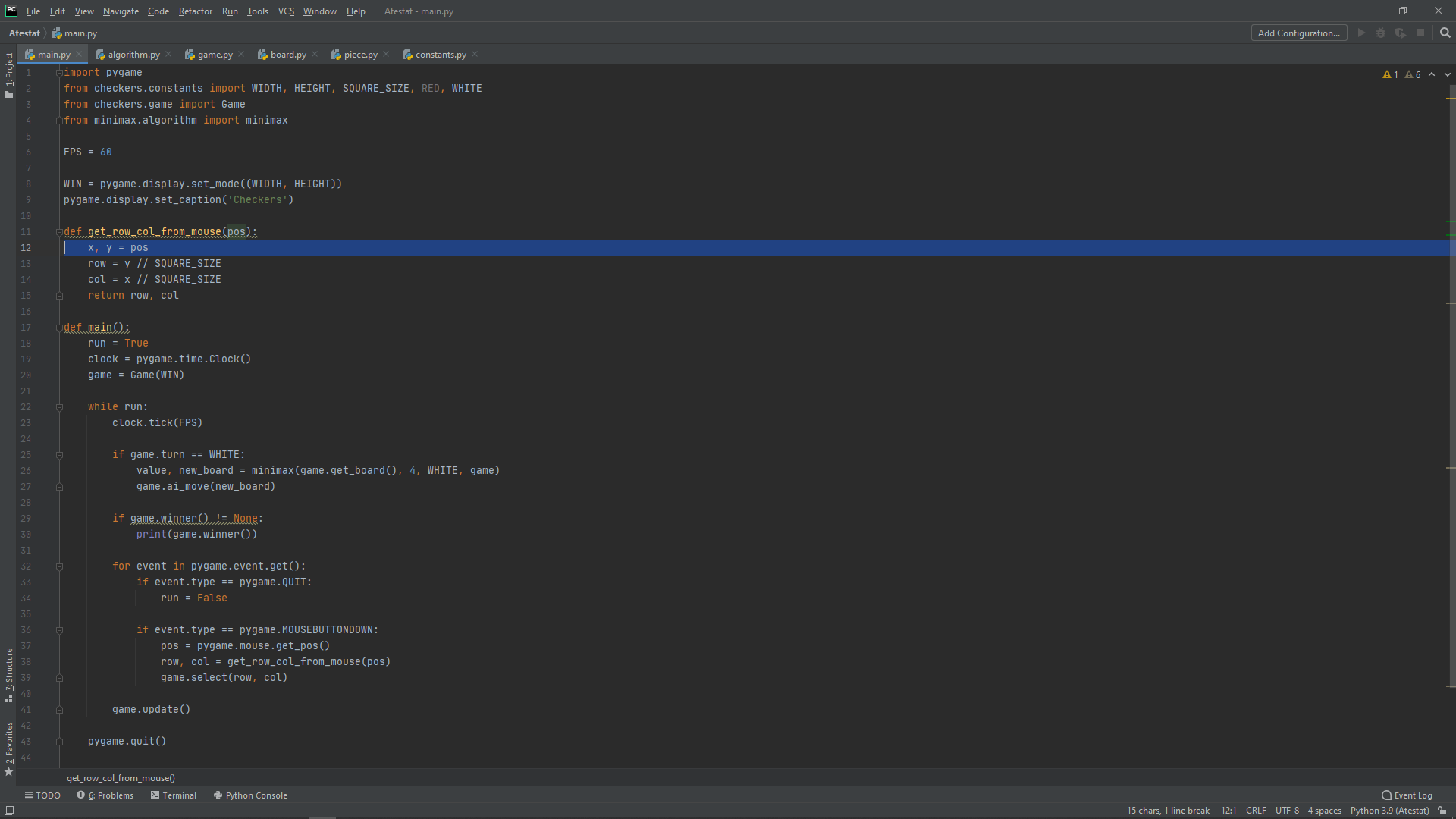
*“Get\_valide\_moves” formeaza un dicționar ce reține pozițiile pe care se poate muta piesa respective.*



*În subprogramele “\_traverse\_left” și “\_traverse\_right”m-am ocupat de pozițiile pe care se pot muta piesele. Aceste funcții furnizează date precum dacă am luat o piesă sau dacă pozițiia pe care o să o avem după ce am luat piesa adversarului este validă.*

def \_traverse\_right(self, start, stop, step, color, right, skipped=[]):  
 moves = {}  
 last = []  
 for r in range(start, stop, step):  
 if right >= COLS:  
 break  
   
 current = self.board[r][right]  
 if current == 0:  
 if skipped and not last:  
 break  
 elif skipped:  
 moves[(r,right)] = last + skipped  
 else:  
 moves[(r, right)] = last  
   
 if last:  
 if step == -1:  
 row = max(r-3, 0)  
 else:  
 row = min(r+3, ROWS)  
 moves.update(self.\_traverse\_left(r+step, row, step, color, right-1,skipped=last))  
 moves.update(self.\_traverse\_right(r+step, row, step, color, right+1,skipped=last))  
 break  
 elif current.color == color:  
 break  
 else:  
 last = [current]  
  
 right += 1  
   
 return moves

## Main()

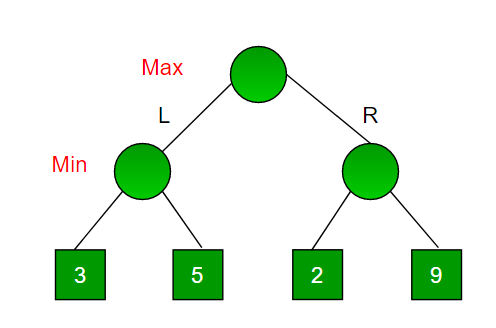
*Main, ca în orice alt limbaj de cod, reprezintă programul principal, unde toate algoritmele creeate până acum sunt unite.* 

# Algoritmul mini-max

## Prezentarea teoretica a algoritmului

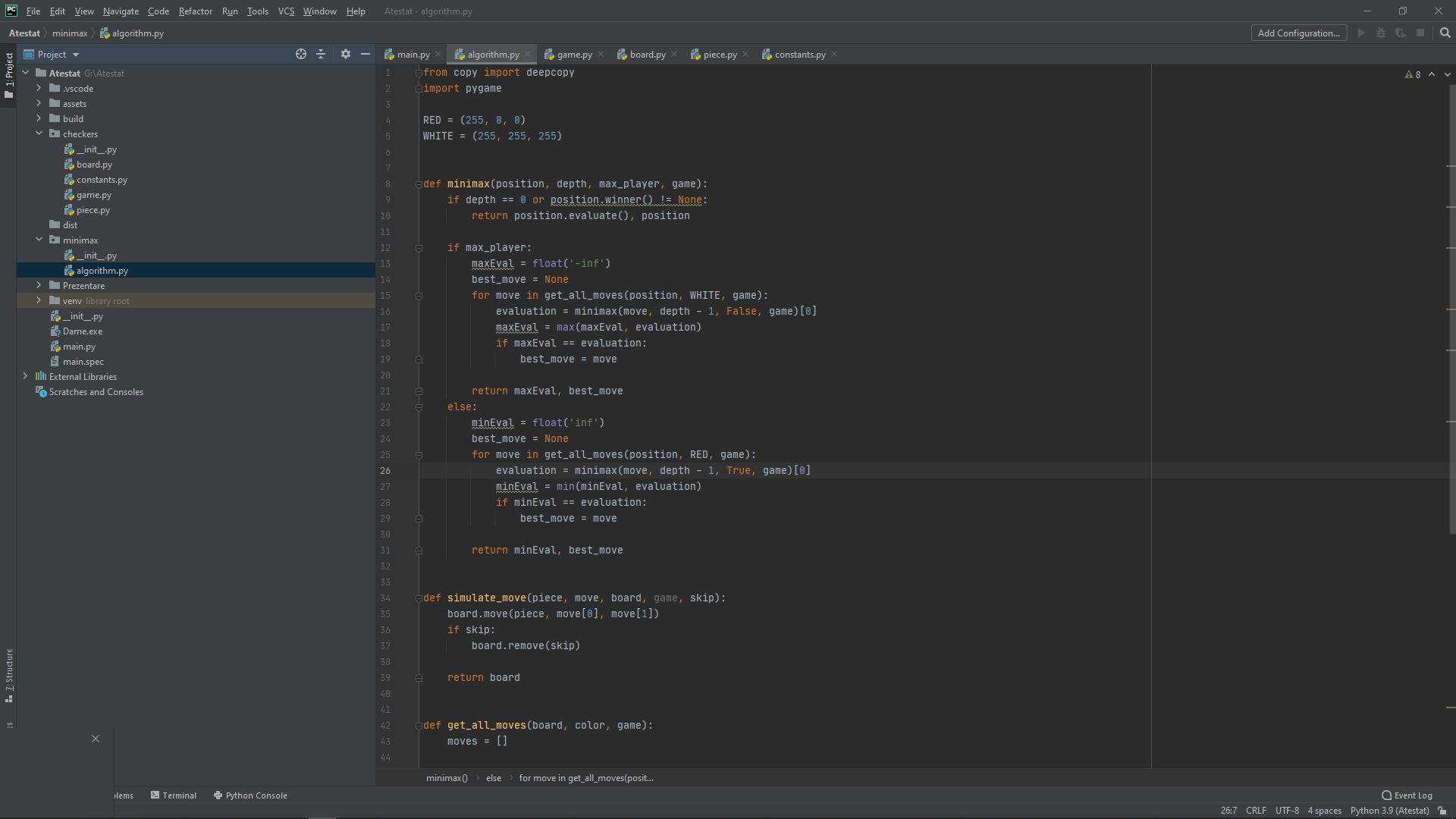
**Minimax** (numit uneori minmax) este o regulă de decizie utilizată în teoria jocurilor, inteligență artificială, statistică și filosofie și care constă în minimizarea pierderii maxime posibile. Alternativ, abordarea poate fi și cea a maximizării câștigului minim (maximin). A început din teoria jocului cu sumă zero cu doi jucători, acoperind atât cazurile în care jucătorii fac mutări alternativ și cele în care fac mutări simultan. Regula a fost extinsă și la jocuri mai complexe și la procese generale de luare a deciziilor în condiții de incertitudine.

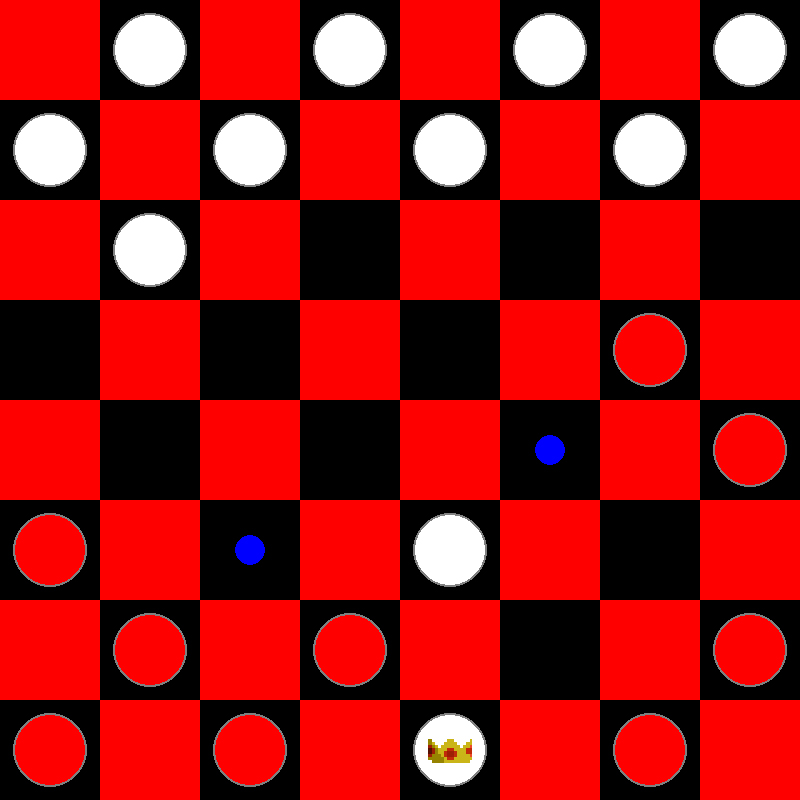
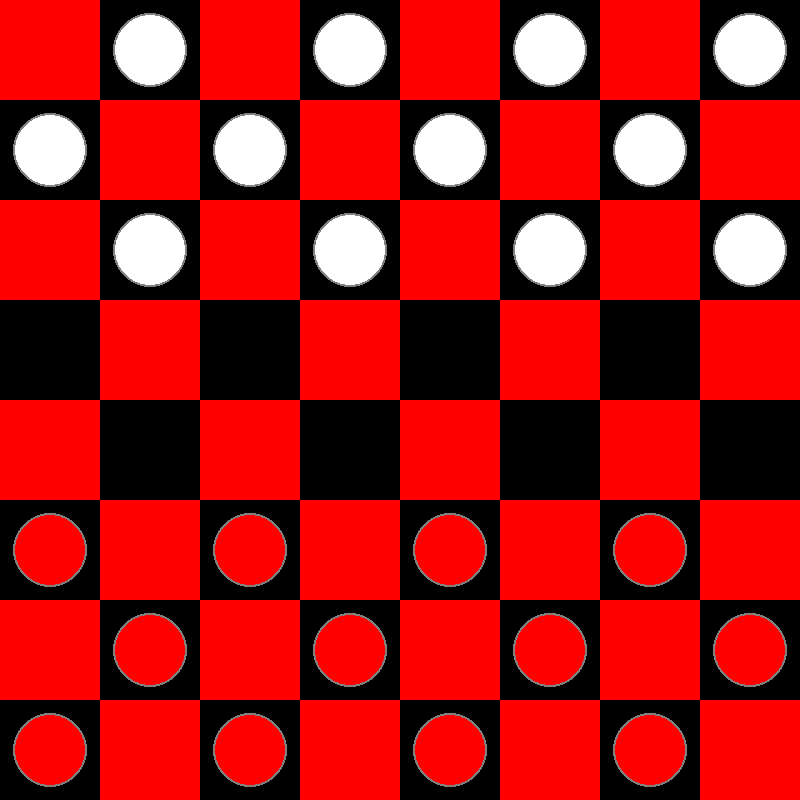
Teoria jocurilor *(în eng. game theory)* este o ramură a matematicii aplicate care abordează problema comportamentului optim în jocurile cu două sau mai multe persoane, într-un cadru descris de un ansamblu de reguli precise care stabilesc posibilitățile de acțiune ale fiecărui jucător, precum și modul cum li se acordă acestora, în final, anumite valori.



## Aplicarea algoritmului minimax în proiect

*Pentru a putea determina mutarea optimă pe care o poate face adversarul am importat librăria copy ce are rolul de a crea o tablă identică de joc cu cea folosită în momentul curent. Pe tabla nouă de joc verificăm pozițiile pe care le poate face adversarul, calculăm riscurile fiecărei poziții și calculam avantajele pentru fiecare mutare. Astfel aflăm poziția preferabilă ce ne va aduce cel mai mare număr de puncte (cele mai multe avantaje).*





# Bibliografie

1. https://www.pygame.org/wiki/about
2. https://www.python.org/about/gettingstarted/
3. https://www.youtube.com/c/TechWithTim/videos
4. https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/
5. https://towardsdatascience.com/how-a-chess-playing-computer-thinks-about-its-next-move-8f028bd0e7b1